



# 3D TRACKING FOR MICROBOONE

Ben Jones



# Introduction

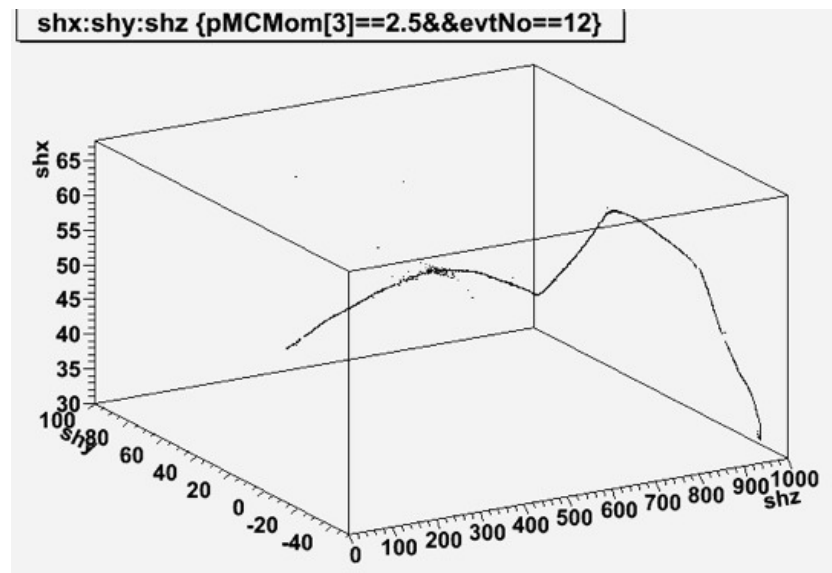
- Been playing with reconstruction ideas on and off at a low effort level for a while
- This “semester” I am on LArSoft at approx 50% effort level – time to make some of this stuff genuinely useful
- Start with trying to produce a robust, analysis-independent 3D reconstruction algorithm
- Progress being made but this is very preliminary discussion, no results yet.

# What a really good tracking algorithm needs

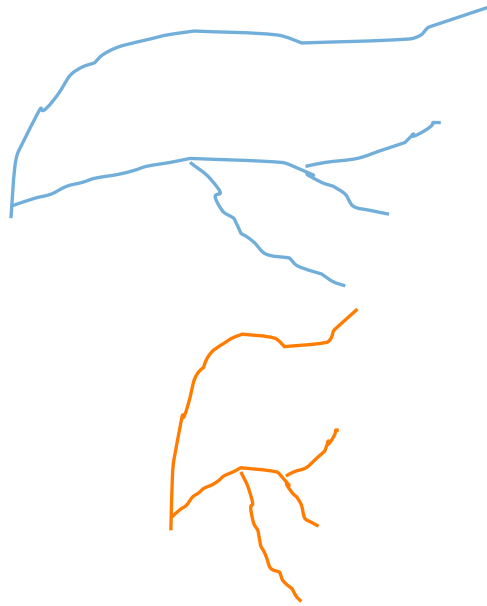
- **1. Curved tracks.** Hough lines are great, but only work with straight-ish tracks (often good for short tracks for this reason)
- **2. Many tracks per event.** Should provide the user with a list of track objects for complicated events
- **3. Track object should contain (or be associated with) all the information the user needs for subsequent PID.** Whether this be at the hit or 3D point level
- **4. Tracks which are joined together should be separable.** This is why we need to take a step beyond clusters.
- **5. Need to make a hypothesis about most likely shape,** not just know all possible shapes (see spacepoint degeneracy problem from Erics studies)
- **6. Track should be able to report back its length, energy measurement in 3 views, total curvature** and possibly lots more.

# The SpacePoint problem

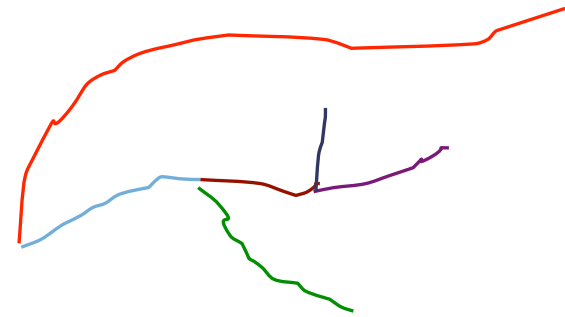
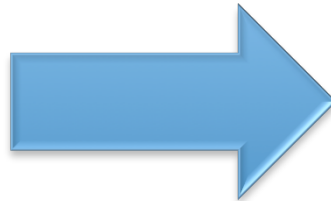
- I have been too harsh on spacepoints. Actually they are incredibly useful, and my algorithm uses them. But it is important to always remember, spacepoints tell you where charge MIGHT BE, not where charge IS.
- However, hits in the 2D view always tell you where charge IS.
- I use spacepoints to seed the track, then fit a 3D object straight onto the 2D hits.



# Scope of this algorithm



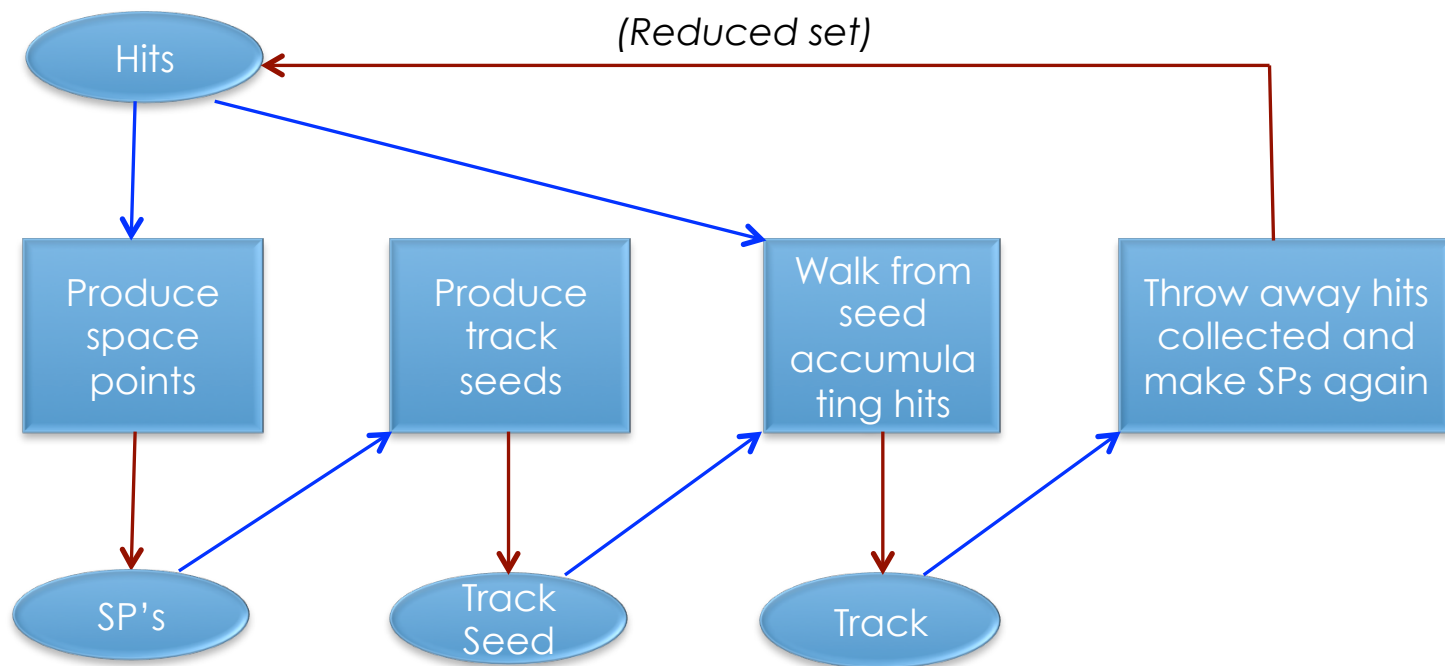
**2D**



**3D**

Provide list of topological sections, with total length, charge, shape information, end points, and then up to next module to event-build from there.

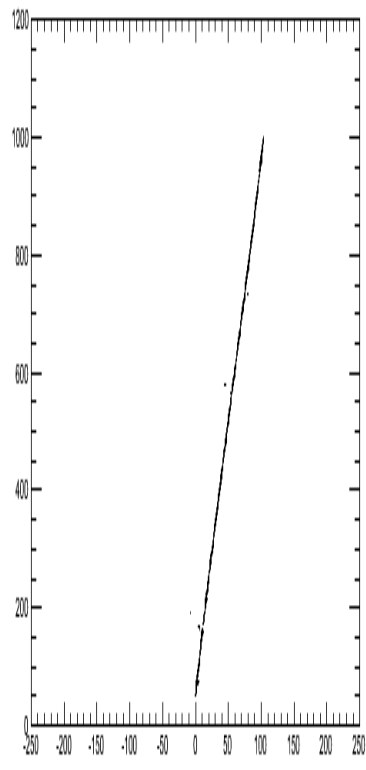
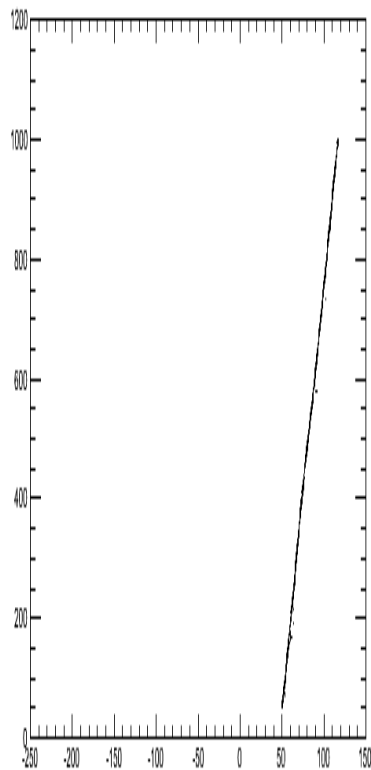
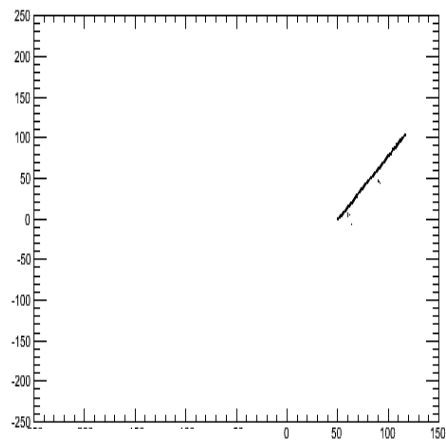
# The “StringFitter” Algorithm



By tracking in 2D, and by acknowledging when hits have already been used, you alleviate spacepoint degeneracies

# 1. Produce space points

- Uses Herbs SpacePointService, and code “borrowed” from Eric to feed his kalman filter.
- Space points are produced from combinations of clusters.
- Spacepoints from one combination of clusters may be one or many tracks.
- I require more than 5 spacepoints from a combination before I bother tracking in it.
- Of my 20 good single muon events, 18 have 1 cluster and 2 have 2 clusters.



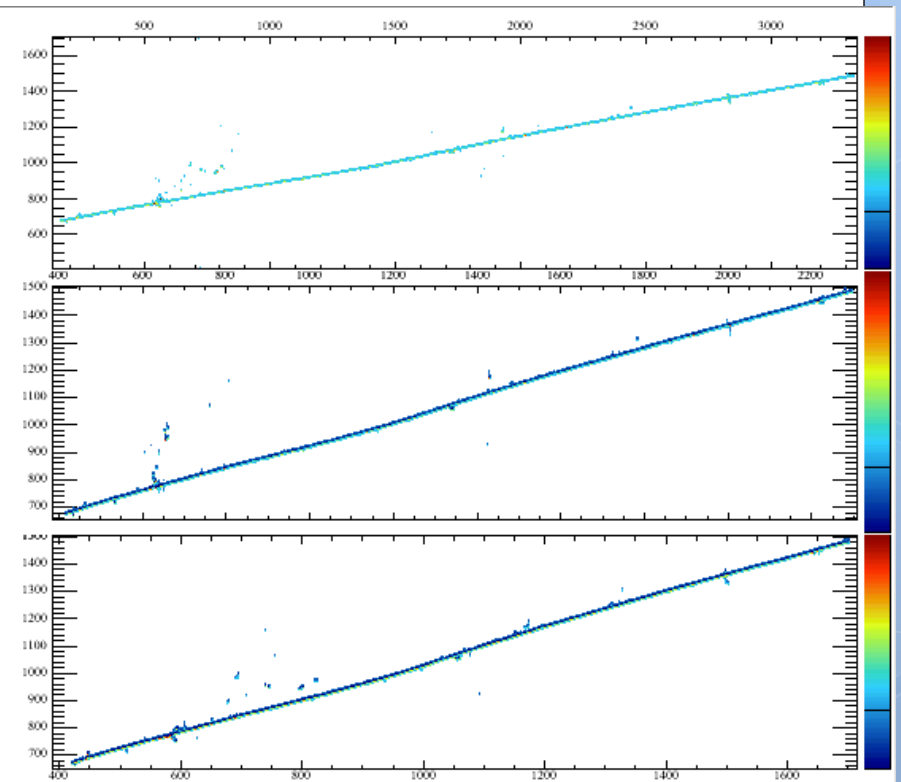
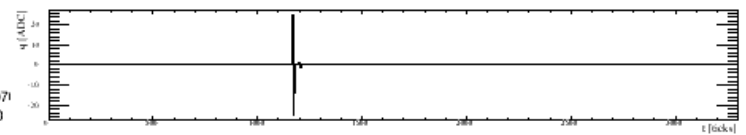
**LArSoft**

Run: 1/0

Event: 6

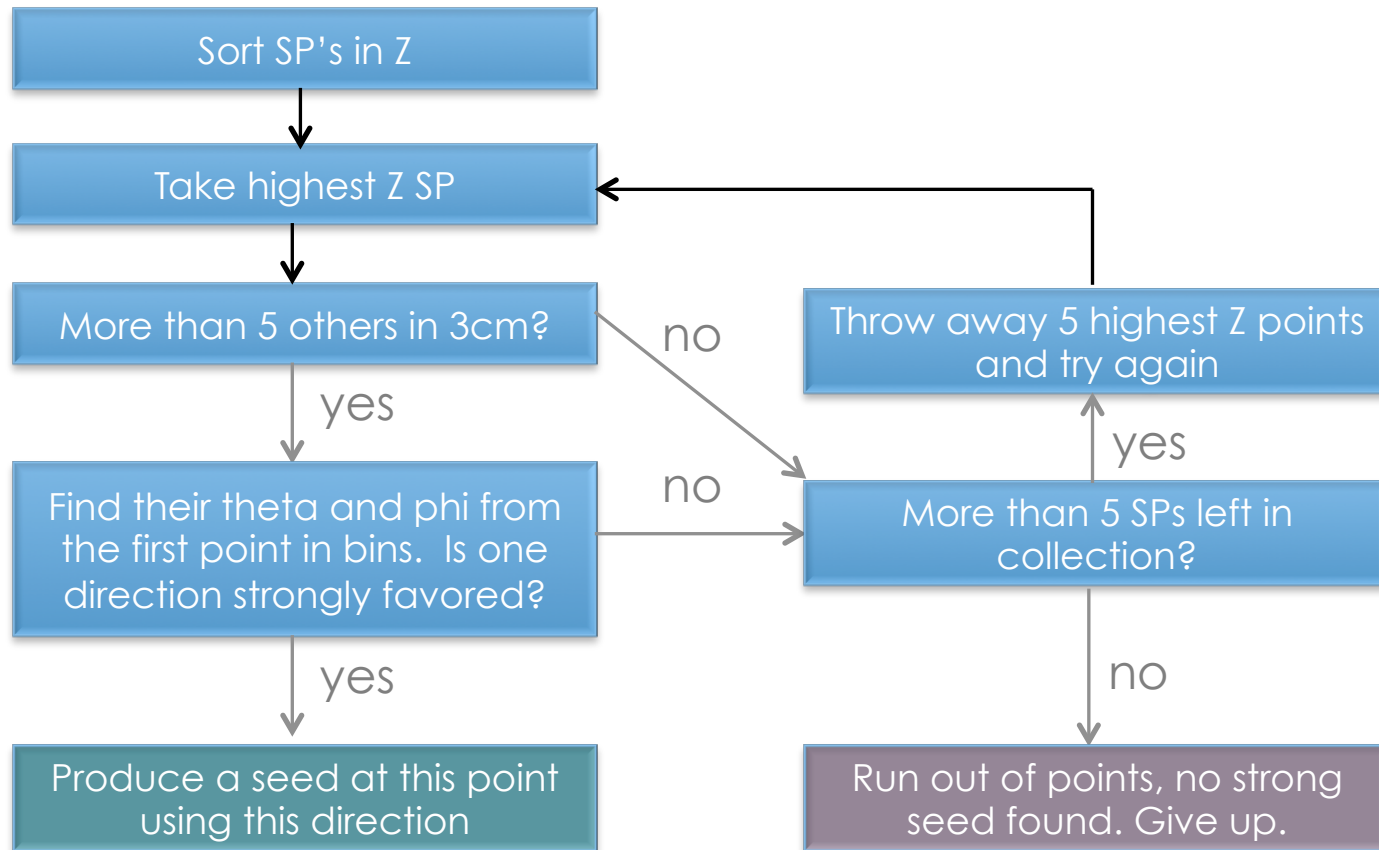
UTC Thu Jan 1, 1971

00:00:0.030000000

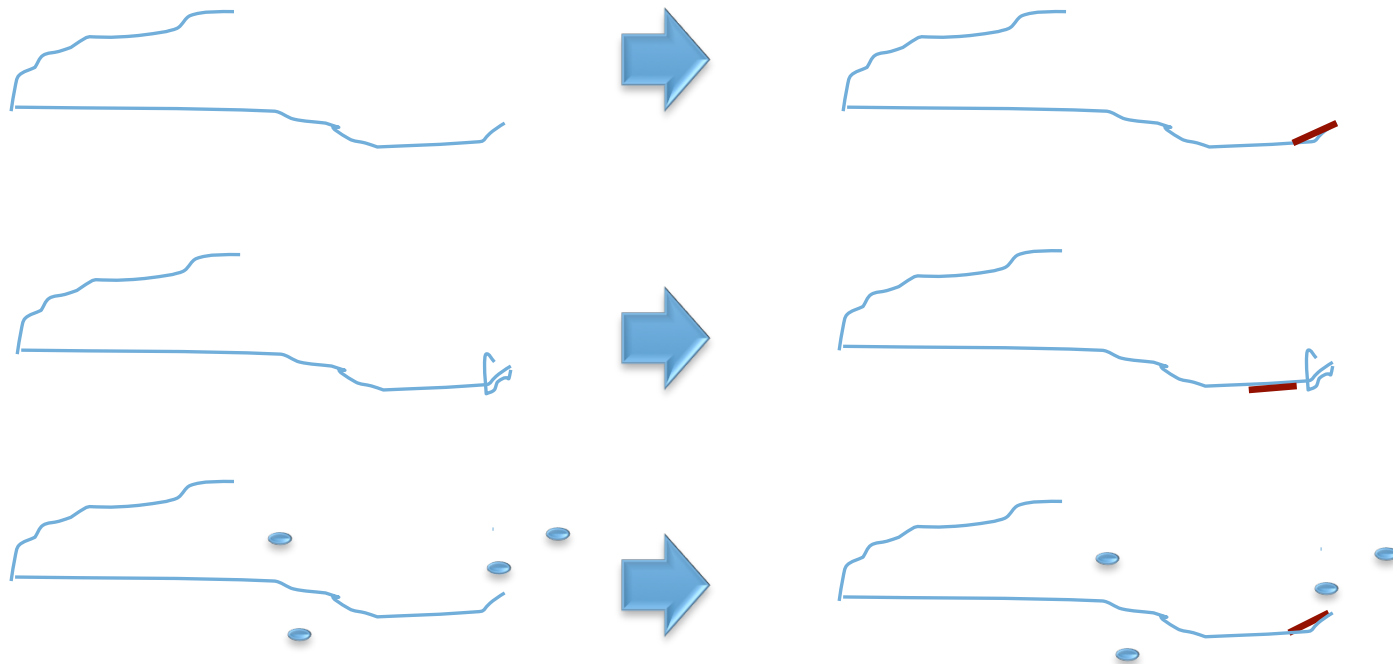


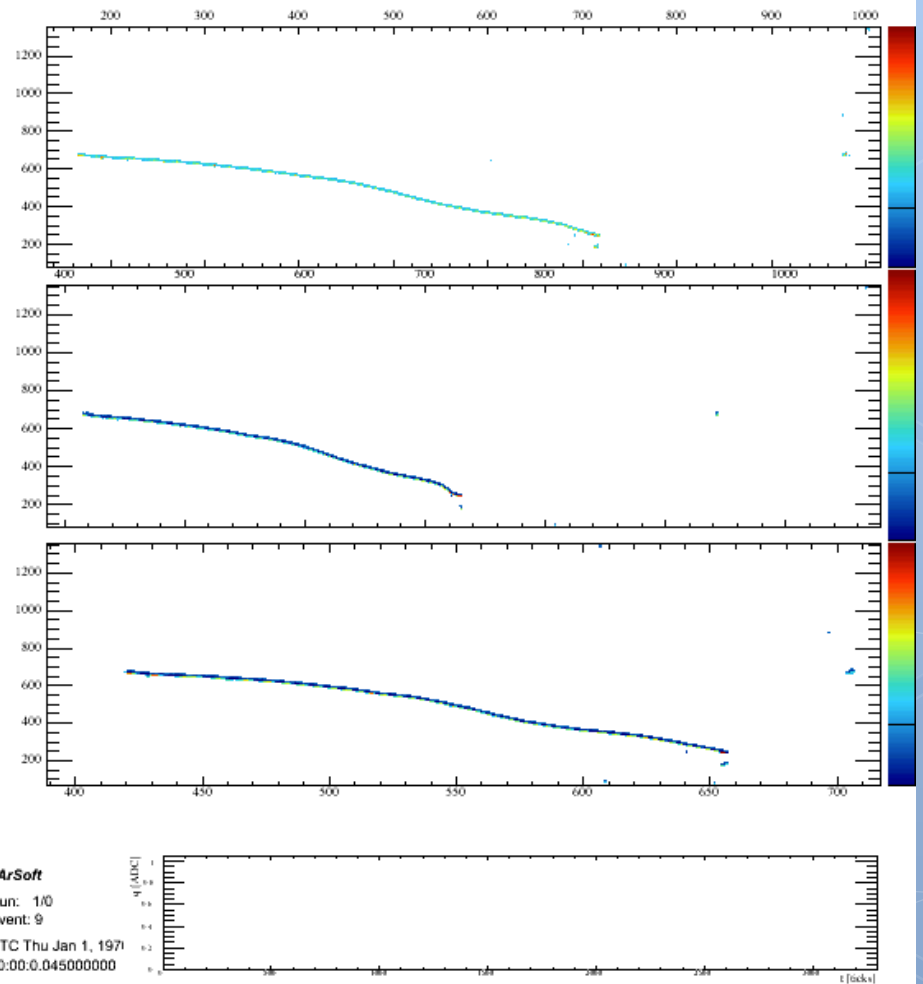
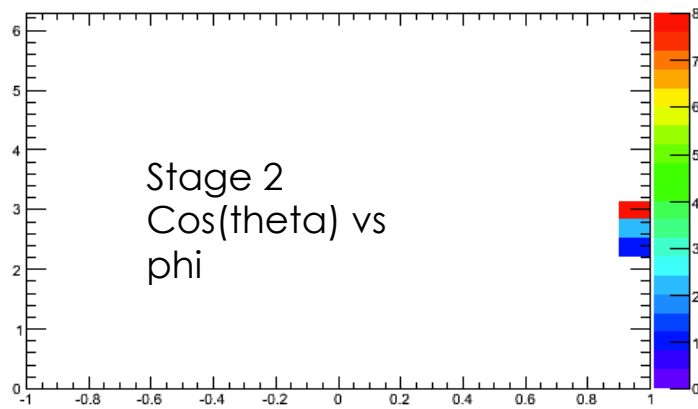
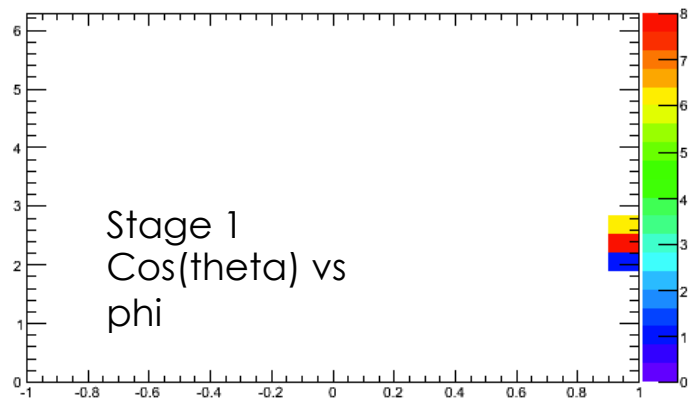


## 2. Produce Seeds



With these rules...





# Status of Seeding

- Seeding seems to work – I am basing this on text output / handscanning.
- Maybe someone can help me make seeds show up on evd? That would be a great way to check if its actually getting it right.
- Also needs testing on some more complicated events (so far just single track muons)

### 3. Stepping along 2D track

- Steps are a finite length (I am working with 2cm).
- One end is fixed at end of last step (or at seed). Angle then free to rotate in 3D space.
- At each step, maximize wrt theta, phi:

$$S = \sum_i \frac{b^2}{(r_i(\theta, \phi) + b)^2} q_i + C \cdot \cos[\theta]$$

Theta constrained to within p/m 15 degrees. If track tries to bend more than this in one step, stop tracing, store track and go again.

Event builder can stick these sections back together if it is useful for a given analysis.

I just made up this function. There may be better ones.  
Mine has :

**2 parameters:**

b : “resolution”

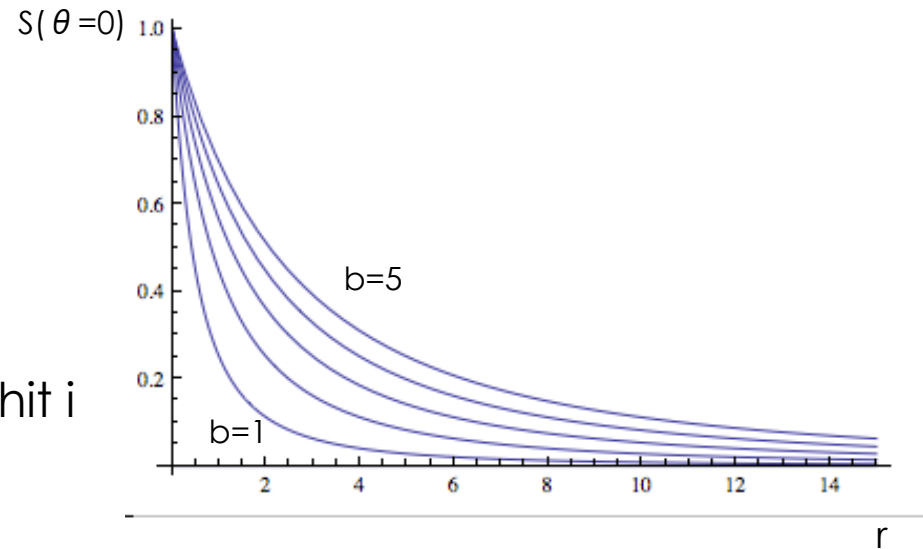
C : track stiffness

**And depends on:**

$q_i$  : charge of hit  $i$

$r_i$  : projected distance of hit  $i$

$\theta$  : angle from previous  
segment

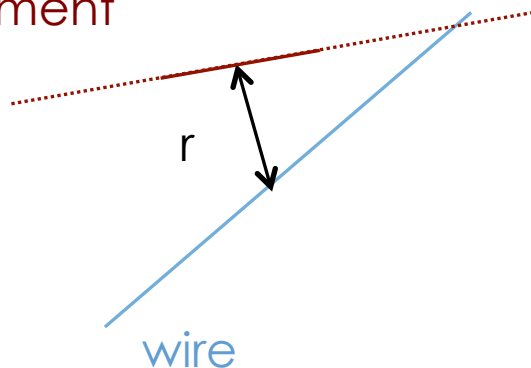


$$S = \sum_i \frac{b^2}{(r_i(\theta, \phi) + b)^2} q_i + C \cdot \cos[\theta]$$

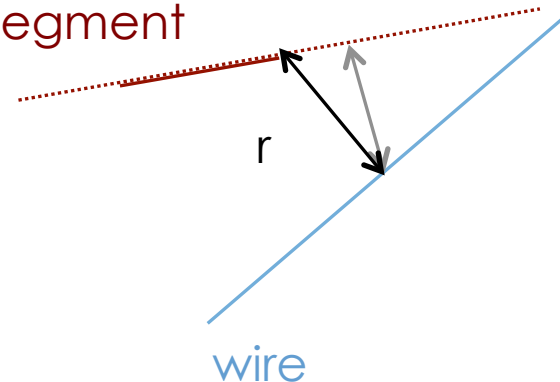
# The projected distance, $r$

- $R$  is the distance of closest approach between the line segment and the 2d hit
- One hit defines a line in 3D space, segment does too, but segment line has ends. Therefore, 2 regimes.

segment

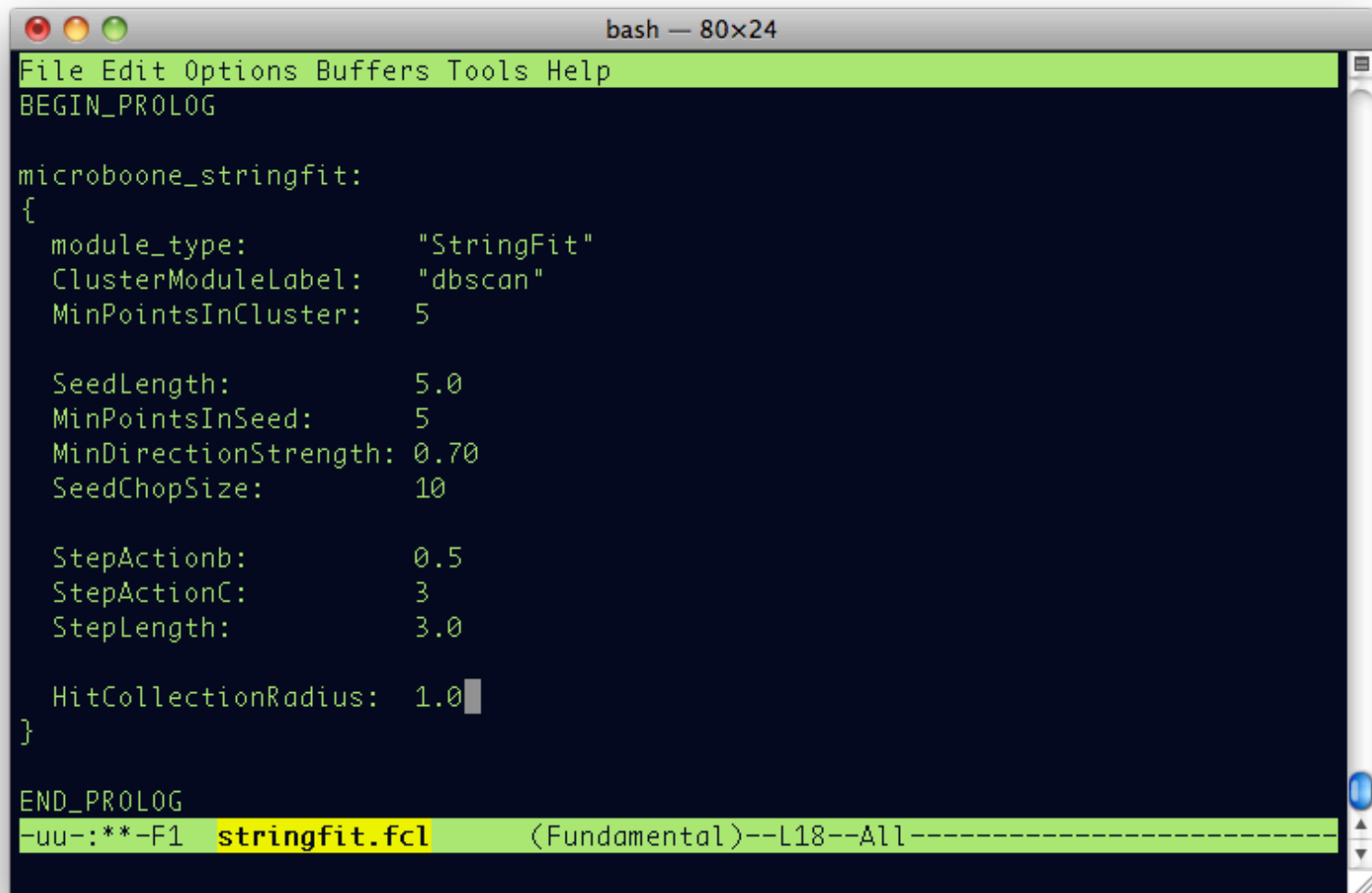


segment



Also, we definitely don't want to loop over every hit for every step. So I first catalogue the hits into maps for easy access, and then for each step use the geometry to preconstraint which wires are close enough to care about before looping over hits.

# Parameters of the Algorithm:



```
bash — 80x24
File Edit Options Buffers Tools Help
BEGIN_PROLOG

microboone_stringfit:
{
  module_type:          "StringFit"
  ClusterModuleLabel:   "dbscan"
  MinPointsInCluster:   5

  SeedLength:           5.0
  MinPointsInSeed:      5
  MinDirectionStrength: 0.70
  SeedChopSize:         10

  StepActionb:          0.5
  StepActionC:          3
  StepLength:           3.0

  HitCollectionRadius:  1.0
}

END_PROLOG
--uu-:**-F1  stringfit.fcl  (Fundamental)--L18--All-----
```



# Status:

Stepping is running and doing things that seem sensible. Needs real testing and validation, which I haven't got to yet.

Also need to produce real output objects to store in event – `recob::Track` with necessary assns to 2d hits. At the moment I just have a `std::vector<TVector3>`

Next stage is to accumulate all hits within some distance of track (1cm?) and throw out, then repopulate spacepoints and seed again. This part should be easy.

Nothing awesome to show today, sorry.